PRELIMINARY

MOSTEK

MD SERIES MICROCOMPUTER MODULES

User's Manual

SYSTEM DESIGN USING THE MOSTEK MD-STD-Z80 BUS

MD-STD-Z80 USER'S MANUAL

Publication Number MK79646

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

SECTION 1

MD-STD-Z80 BUS DESCRIPTION

The purpose of the MD-STD-Z80 user's manual is to provide the O.E.M. system designer with detailed information about the Mostek MD-STD-Z80 BUS. The information presented is a bus description and bus timing for the MD-STD-Z80 BUS.

The MD-STD-Z80 BUS was defined by Mostek exclusively for the Z80 as a subset of the general purpose STD BUS. The general purpose STD BUS was defined for the Z80, 8085, 6800, and 6809, but does not include exact functional pin descriptions and bus timing.

It was for these reasons that the MD-STD-Z80 BUS was defined. Therefore, by defining the system exclusively for the Z80, exact functional pin descriptions and bus timing, can be given so that all boards designed to the MD-STD-Z80 specification will work together.

For users interested in software information, the following MOSTEK manuals should be consulted: Z80 PROGRAMMING MANUAL, MK78515
Z80 MICRO-REFERENCE MANUAL, MK78516
MOSTEK MICROCOMPUTER DATA BOOK, MK79707

TABLE 1-1

MD-STD-Z80 BUS DESCRIPTION

BUS PIN	MNEMONIC	DESCRIPTION
1	+5V	+5Vdc system power
2	+5V	+5Vdc system power
3	GND	Ground-System signal ground and DC return
4	GND	Ground-System signal ground and DC return
5	-5V	-5Vdc system power
6	-5V	-5Vdc system power
7	D3	Data Bus (Tri-state, input/output, active high).
8	D7	D0-D7 constitute an 8-bit bidirectional data
9	D2	bus. The data bus is used for data exchange
10	D6	with memory and I/O devices.
11	D1	
12	D5	
13	D0	
14	D4	
15	A7	Address Bus (Tri-state, output, active high).
16	A15	A0-A15 make up a 16-bit address bus. The ad-
17	A6	dress bus provides the address for memory (up to
18	A14	65K bytes) data exchanges and for I/O device
19	A5	data exchanges. I/O addressing uses the lower 8
20	A13	address bits to allow the user to directly
21	A4	select up to 256 input or 256 output ports. A0
22	A12	is the least significant address bit. During
23	A3	refresh time, the lower 7 bits contain a valid
24	A11	refresh address for dynamic memories.
25	A2	
26	A10	
27	A1	

BUS PIN	MNEMONIC	DESCRIPTION
28	A9	
29	A0	
30	A8	
31	/WR	Write (Tri-state,active low). /WR indicates that the CPU data bus holds valid data to be stored in the addressed memory or I/O device.
32	/RD	Read (Tri-state, output, active low). /RD indicates that the CPU wants to read data from memory or an I/O device. The addressed I/O device or memory should use this signal to gate data onto the CPU data bus.
33	/IORQ	Input/Output Request (Tri-state, output, active low). The /IORQ signal indicates that the lower half of the address bus holds a valid I/O address for an I/O read or write operation. An /IORQ signal is also generated with an /M1 signal when an interrupt is being acknowledged to indicate that an interrupt response vector can be placed on the data bus. Interrupt Acknowledge operations occur during /M1 time, while I/O operations never occur during /M1 time.
34	/MEMRQ	Memory Request (Tri-state,output, active low). The /MEMRQ signal indicates that the address bus holds a valid address for a memory read or memory write operation.

BUS PIN	MNEMONIC	DESCRIPTION
35	/IOEXP	I/O Expansion, not used on Mostek MD cards. (Normally strapped to ground on the MOSTEK motherboard)
36	/MEMEX	Memory Expansion, not used on Mostek MD cards. (Normally strapped to ground on the MOSTEK motherboard)
37	/REFRESH	REFRESH (Tri-state, output, active low). /REFRESH indicates that the lower 7 bits of the address bus contain a refresh address for dynamic memories and the /MEMRQ signal should be used to perform a refresh cycle for all dynamic RAMs in the system. During the refresh cycle, A7 is a logic 0 and the upper 8 bits of the address bus contain the I register.
38	/MCSYNC	Not generated by the MOSTEK CPU cards. /MCSYNC can be generated by gating the following signals: /RD + /WR + /INTAK. By connecting a jumper on the MDX-CPU, this line becomes /DEBUG (Input). /DEBUG is used in conjuction with the DDT-80 operating system on the MD-DEBUG card and the MD-SST card for implementing a hardware single step function. When pulled low, the /DEBUG line will set an address modification latch which will force the upper three address lines A15, A14, and A13 to a logic 1. These address lines will remain at a logic 1 until reset by performing any I/O operation.

BUS PIN	MNEMONIC	DESCRIPTION
39	/STATUS 1 (M1)	Machine Cycle One. (Tri-state, output, active low). /M1 indicates that the current machine cycle is in the op code fetch cycle of an instruction. Note that during the execution of two byte opcodes, /M1 will be generated as each op code is fetched. These two byte opcodes always begin with a CBh, DDh, EDh, or FDh. /M1 also occurs with /IORQ to indicate an interrupt acknowledge cycle.
40	/STATUS 0	Not used on Mostek MD cards.
41	/BUSAK	Bus Acknowledge (Output, active low). Bus acknowledge is used to indicate to the requesting device that the CPU address bus, data bus, and control bus signals have been set to their high impedance state and the external device can now control the bus.
42	/BUSRQ	Bus Request (Input, active low). The /BUSRQ signal is used to request the CPU address bus, data bus, and control signal bus to go to a high impedance state so that other devices can control those buses. When /BUSRQ is activated, the CPU will set these buses to a high impedance state as soon as the current CPU machine cycle is terminated and the /BUSAK signal is activated.
43	/INTAK	Interrupt Acknowledge (Tri-state, output, active low). The /INTAK signal indicates that an

BUS PIN	MNEMONIC	DESCRIPTION
		interrupt acknowledge cycle is in progress, and the interrupting device should place its response vector on the data bus. The /INTAK signal is equivalant to an /IORQ during an /M1.
44	/INTRQ	Interrupt Request (Input, active low). The Interrupt Request signal is generated by I/O devices. A request will be honored at the end of the current instruction if the internal software controlled interrupt enable flip flop (IFF) is enabled and if the /BUSRQ signal is not active. When the CPU accepts the interrupt, an interrupt acknowledge signal /INTAK (/IORQ during an /M1) is sent out at the beginning of the next instruction.
45	/WAITRQ	Wait Request (Input, active low). Wait Request indicates to the CPU that the addressed memory or I/O device is not ready for a data transfer. The CPU continues to enter wait states for as long as this signal is active. This signal allows memory or I/O devices of any speed to be synchronized to the CPU. Use of this signal postpones refresh as long as it is held active.
46	/NMIRQ	Non Maskable Interrupt Request (Input, negative edge triggered). The Non Maskable Interrupt Request line has a higher priority than the /INTRQ line and is always recognized at the end of the current instruction, independent of the status of the interrupt enable flip-flop.

BUS PIN	MNEMONIC	DESCRIPTION
		/NMIRQ automatically forces the CPU to restart to location 0066h. The program counter is automatically saved in the external stack so that the user can return to the program that was interrupted. Note that continuous WAIT cycles can prevent the current instruction from ending, and that a /BUSRQ will override a /NMIRQ.
47	/SYSRESET	System Reset (Output, active low). The System Reset line indicates that a reset has been generated either from an external reset or the power on reset circuit. The system reset will occur only once per reset request and will be approximately 2us in duration. A system reset will also force the CPU program counter to zero, disable interrupts, set the I register to 00h, set the R register to 00h, and set Interrupt Mode 0.
48	/PBRESET	Push Button Reset (Input, active low). The Push Button Reset will generate a debounced system reset.
49	/CLOCK	Processor Clock (Output, active low.) Single phase system clock.
50	/CNTRL	Not used on MOSTEK MD cards.
51	/PCO	Priority Chain Output (Output, active high) This signal is used to form a priority interrupt

BUS PIN	MNEMONIC	DESCRIPTION
		daisy chain when more than one interrupt driven device is being used. A high level on this pin indicates that no other devices of higher priority are being serviced by a CPU interrupt service routine.
52	PCI	Priority Chain In (Input, active high). This signal is used to form a priority interrupt daisy chain when more than one interrupt driven device is being used. A high level on this pin indicates that no other devices of higher priority are being serviced by a CPU interrupt service routine.
53	AUX GND	Auxiliary Ground (Bussed)
54	AUX GND	Auxiliary Ground (Bussed)
55	+12V	+12Vdc system power
56	-12V	-12Vdc system power

NOTES:

1. Input/Output references of each signal are made with respect to MDX-CPU module.

2. The following signals have pull-up resistors: /WR, /RD, /IORQ, /MEMRQ, /REFRESH, /DEBUG, /M1, /BUSRQ, /INTAK, /INTRQ, /WAITRQ, /NMIRQ, /SYSRESET, /PBRESET and /CLOCK.

TABLE 1-2

MD-STD-Z80 BUS PIN-OUT

COMPONENT SIDE		CIRCUIT SIDE	
PIN	MNEMONIC	PIN	MNEMONIC
1	+5V	2	+5V
3	GND	4	GND
5	-5V	6	-5V
7	D3	8	D7
9	D2	10	D6
11	D1	12	D5
13	D0	14	D4
15	A7	16	A15
17	A6	18	A14
19	A5	20	A13
21	A4	22	A12
23	A3	24	A11
25	A2	26	A10
27	A1	28	A9
29	A0	30	A8
31	/WR	32	/RD
33	/IORQ	34	/MEMRQ
35	/IOEXP	36	/MEMEX
37	/REFRESH	38	/MCSYNC
39	/STATUS 1	40	/STATUS 0
41	/BUSAK	42	/BUSRQ
43	/INTAK	44	/INTRQ
45	/WAITRQ	46	/NMIRQ
47	/SYSRESET	48	/PBRESET
49	/CLOCK	50	/CNTRL
51	PC0	52	PCI
53	AUX GND	54	AUX GND
55	+12V	56	-12V

STD-Z80 ELECTRICAL BUS SPECIFICATIONS

BUS RECEIVERS

Logical Low: 0.8V maximum at -0.36mA
Logical High: 2.0V minimum at 20uA

BUS DRIVERS

Logical Low: 0.5V maximum at 24MA
Logical High:2.4V mimimum at -15MA
Off State Output Current (tri-state): ± 100uA

RECOMMENDED BUS DRIVERS AND RECEIVERS

Bus drivers: 74LS240, 74LS241, 74LS373, 74LS374, and 74LS244.

Bus Receivers: 74LS240, 74LS241, and 74LS244.

Bus Transceivers: 74LS245, 74LS242, and 74LS243.

SECTION 2
BUS TIMING

INTRODUCTION

This section gives the BUS timing for the MD-CPU at 2.5 MHz and at 4.0 MHz. However, it should be noted that the central timing element on the MD-CPU is the Z80 (MK3880). For further information on timing, or operating modes, the user should consult the Z80 DATA BOOK which is availible from MOSTEK.

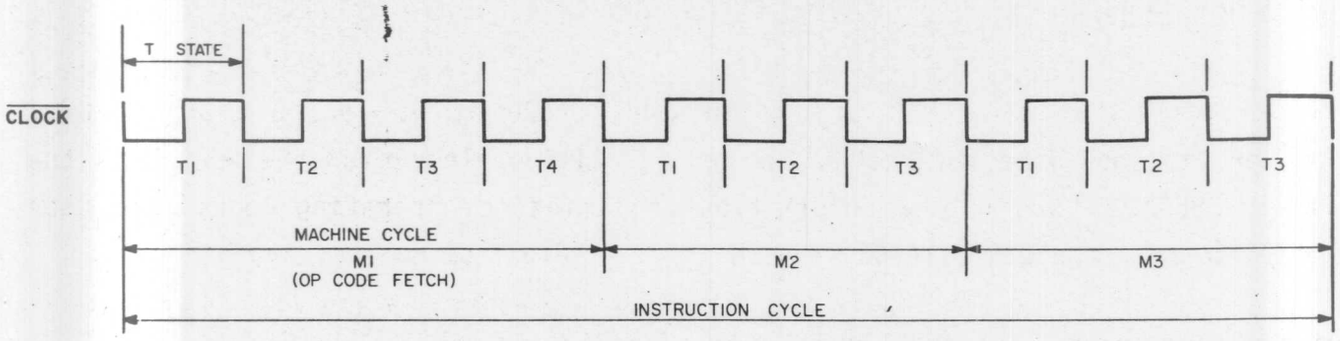
MD-CPU BUS TIMING

The MD-CPU executes instructions by stepping through a very precise set of a few basic operations. Theses include:

- Memory read or write
- I/O device read or write
- Interrupt acknowledge

All instructions are merely a series of these basic operations. Each of these basic operations can take from three to six clock periods to complete or they can be lengthened to synchronize the CPU to the speed of external devices. The basic clock periods are referred to as T states and the basic operations are referred to as M (for machine) cycles. Figure 2-1 illustrates how a typical instruction will be merely a series of specific M cycles and T states. Notice that this instruction consists of three machine cycles (M1, M2 and M3). The first machine cycle of any instruction is a fetch cycle which is four, five or six T states long (unless lengthened by the wait signal). The fetch cycle (M1) is used to fetch the OP code of the next instruction to be executed. Subsequent machine cycles move data between the CPU and memory or I/O devices and they may have anywhere from three to five T cycles (again they may be lengthened by wait states to synchronize the external devices to the CPU). The following paragraphs describe the timing which occurs within any of the basic machine cycles.

FIGURE 2-1 BASIC CPU TIMING EXAMPLE



All CPU timing can be broken down into a few very simple timing diagrams as shown in Figure 2-2 through 2-17 . These diagrams show the following basic operations with and without wait states (wait states are added to synchronize the CPU to slow memory or I/O devices). A complete machine cycle breakdown is given in section 3.

INSTRUCTION FETCH

Figure 2-2 shows the timing during an M1 cycle (OP code fetch). Notice that the PC is placed on the address bus at the beginning of the M1 cycle. One half clock time later the /MEMRQ signal goes active. At this time the address to the memory has had time to stabilize so that the falling edge of /MEMRQ can be used directly as a chip enable for edge activated memories. The /RD line also goes active to indicate that the memory read data should be enabled onto the data bus. The CPU samples the data from the memory on the data bus with the falling edge of the clock of state T3 and this same edge is used by the CPU to turn off the /RD and /MEMRQ signals. Thus the data has already been sampled by the CPU before the /RD signal becomes inactive. Clock state T3 and T4 of a fetch cycle are used to refresh dynamic memories. (The CPU uses this time to decode and execute the fetched instruction so that no other operation could be performed at this time). During T3 and T4 the lower 7 bits of the address bus contain a memory refresh address and the /REFRESH signal becomes active to indicate that a refresh cycle for all dynamic memories should be performed. Notice that a /RD signal is not generated during refresh time to prevent data from different memory segments from being gated onto the data bus. The /MEMRQ signal during refresh time should be used to start the refresh cycle since the refresh address is only guaranteed to be stable during /MEMRQ time.

FIGURE 2-2 INSTRUCTION FETCH

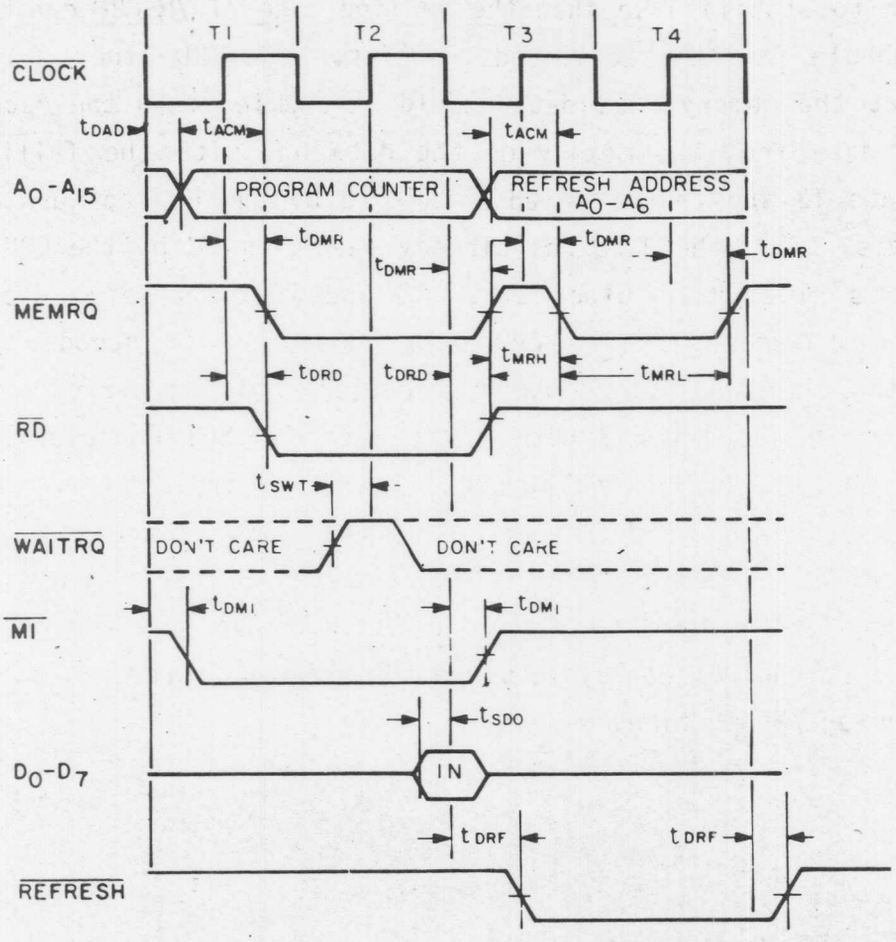
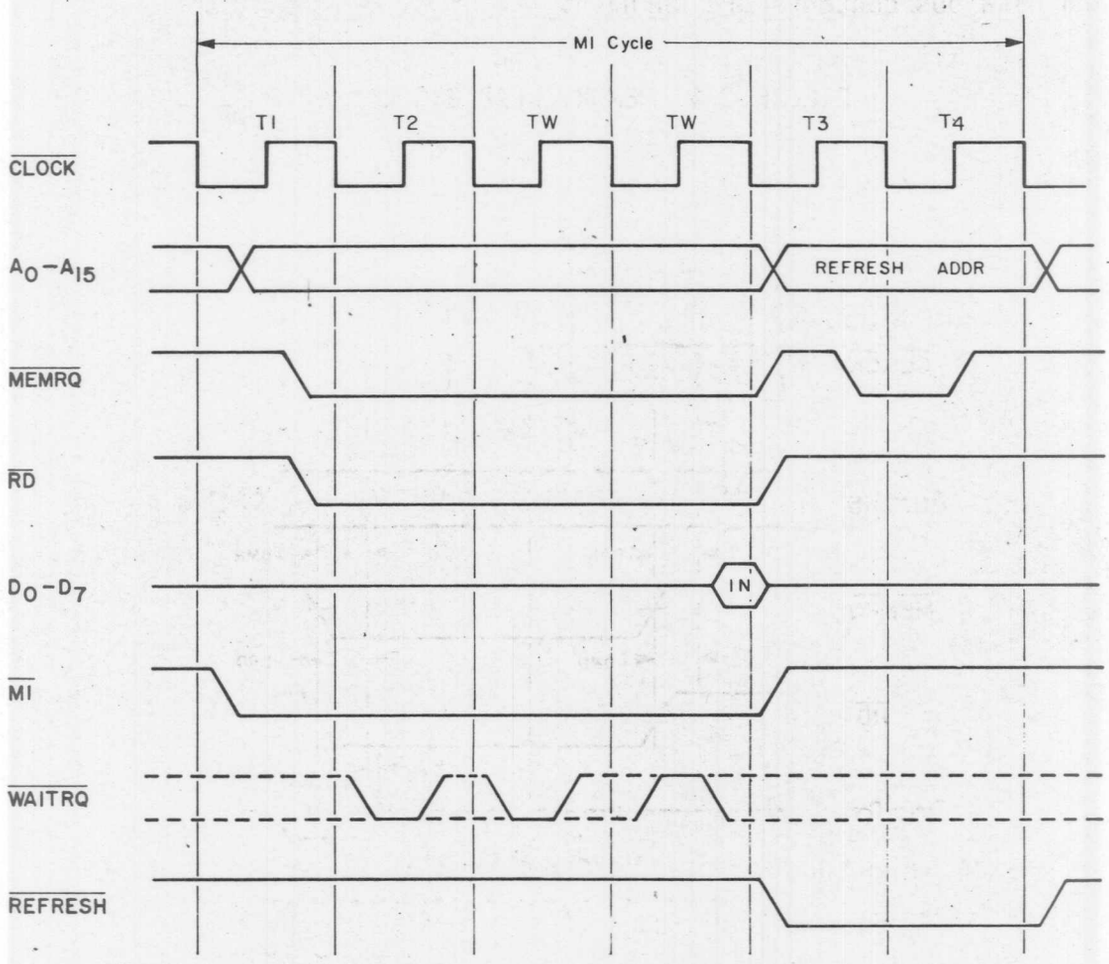


Figure 2-3 illustrates how the fetch cycle is delayed if the memory activates the /WAITRQ line. During T2 and every subsequent Tw, the CPU samples the /WAITRQ line with the rising edge of /CLOCK. If the /WAITRQ line is active at this time, another wait state will be entered during the following cycle. Using this technique the read cycle can be lengthened to match the access time of any type of memory device.

FIGURE 2-3 INSTRUCTION OP CODE FETCH WITH WAIT STATES



MEMORY READ OR WRITE

Figures 2-4 and 2-5 illustrate the timing of memory read or write cycles other than an OP code fetch (M1 cycle). These cycles are generally three clock periods long unless wait states are requested by the memory via the /WAITRQ signal. The /MEMRQ signal and the /RD signal are used the same as in the fetch cycle. In the case of a memory write cycle, the /MEMRQ also becomes active when the address bus is stable. The /WR line is active when data on the data bus is stable so that it can be used directly as a R/W pulse for virtually any type of semiconductor memory. Furthermore the /WR signal goes inactive one-half T state before the address and data bus contents are changed.

FIGURE 2-4 MEMORY READ CYCLE

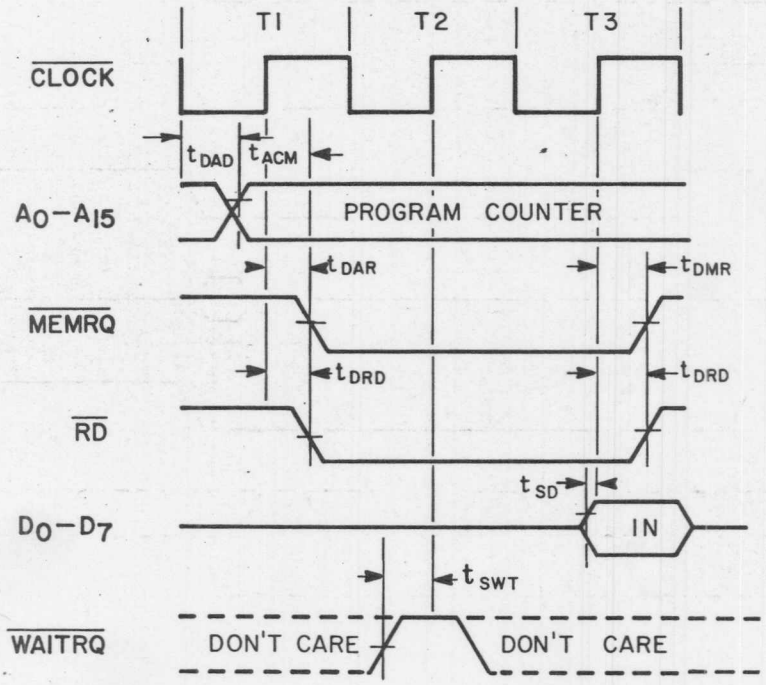


FIGURE 2-5 MEMORY WRITE CYCLE

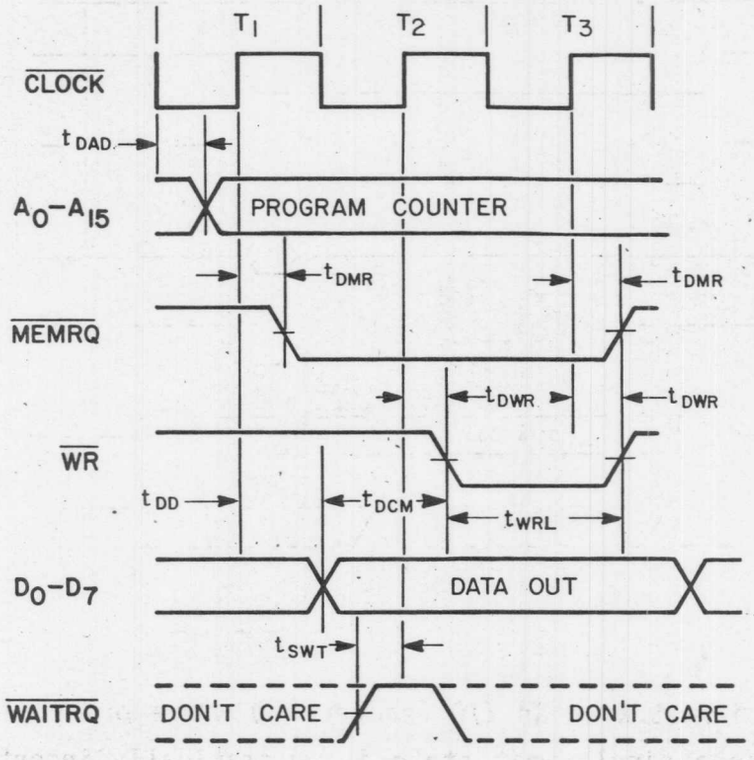
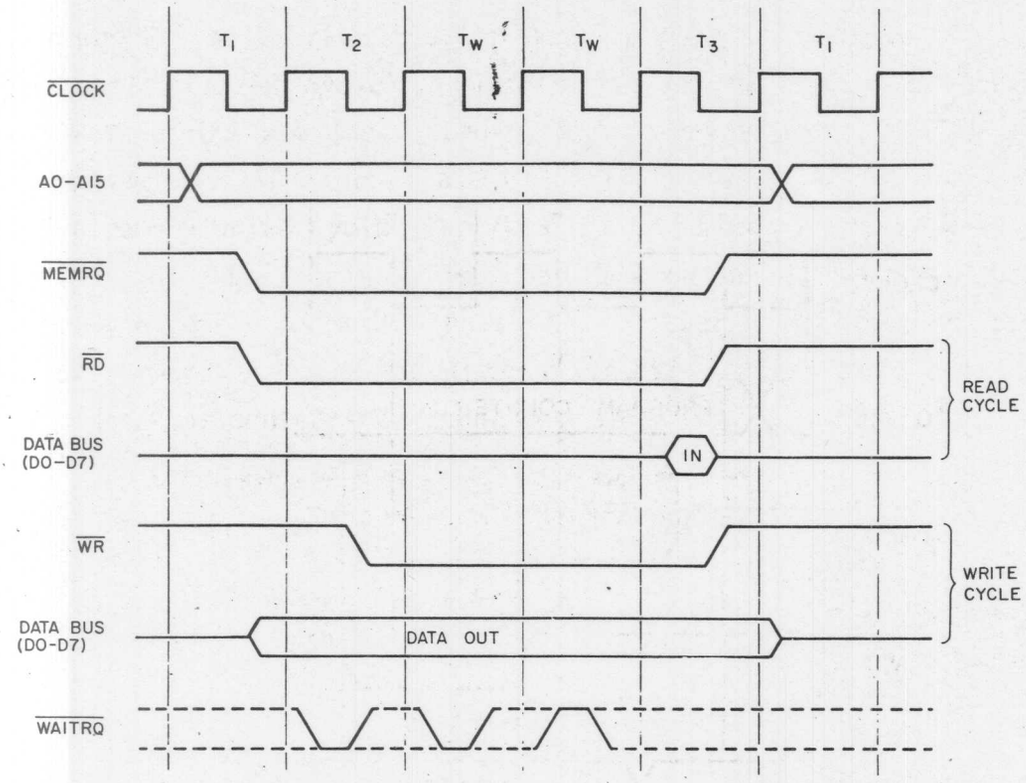


Figure 2-6 illustrates how a /WAITRQ signal will lengthen any memory read or write operation. This operation is identical to that previously described for a fetch cycle. Notice in this figure that a seperate read and separate write cycle are shown in the same figure although read and write cycles can never occur simultaneously.

FIGURE 2-6 MEMORY READ OR WRITE CYCLES WITH WAIT STATES



INPUT OR OUTPUT CYCLES

Figures 2-7 and 2-8 illustrates an I/O read or I/O write operation. Notice that during I/O operations a single wait state is automatically inserted. The reason is that during I/O operations, the time from the /IORQ signal until the time that the CPU must sample the /WAITRQ line is very short and without this extra state, sufficient time does not exist for an I/O port to decode its address and activate the /WAITRQ line if a wait is required. During a read I/O operation, the /RD line is used to enable the addressed port onto the data bus just as in the case of a memory read. For I/O write operations, the /WR line is used as a clock to the I/O port, again with sufficient overlap timing automatically provided so that the rising edge may be used as a data clock.

Figure 2-9 illustrates how additional wait states may be added with the /WAITRQ line.

FIGURE 2-7 I/O READ CYCLE

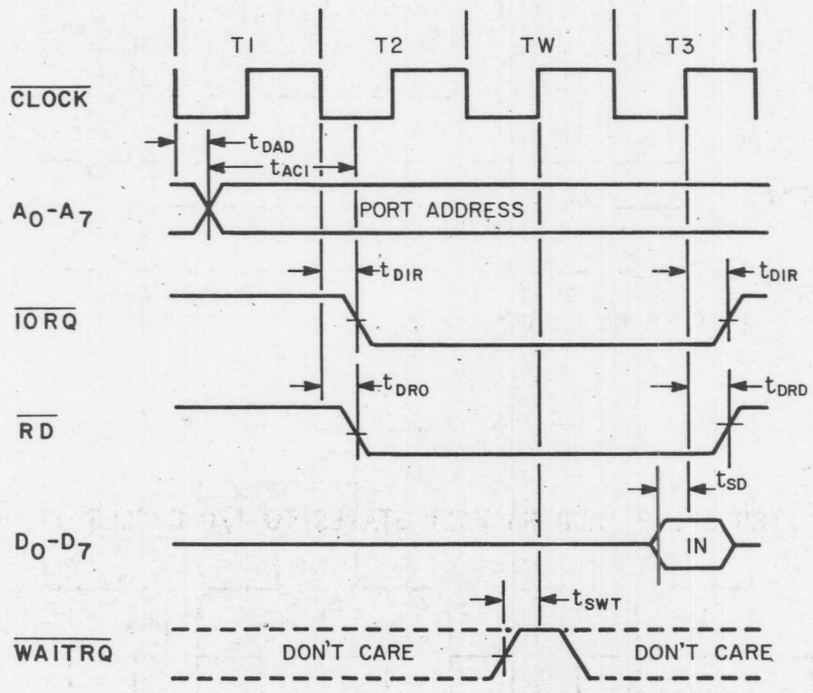


FIGURE 2-8 I/O WRITE CYCLE

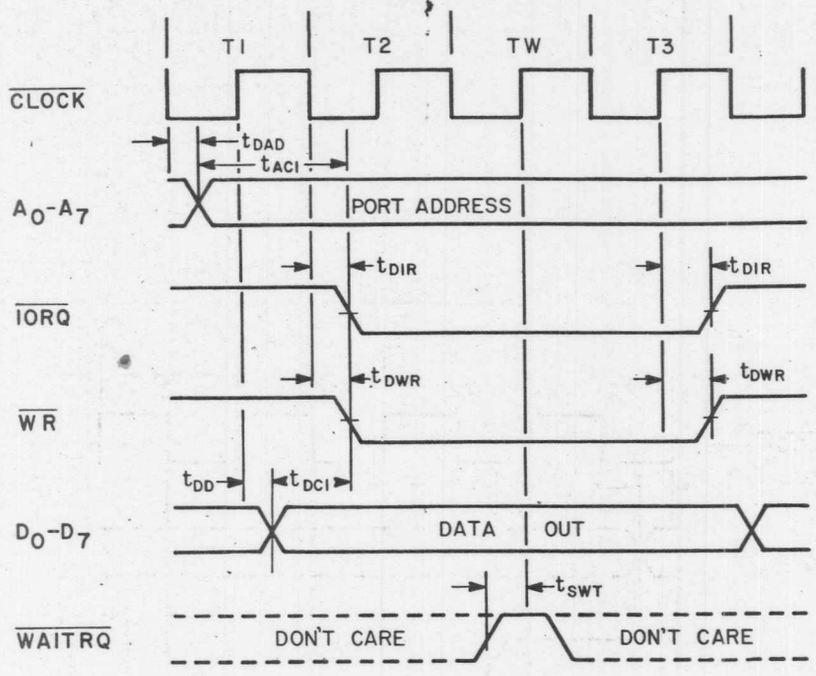
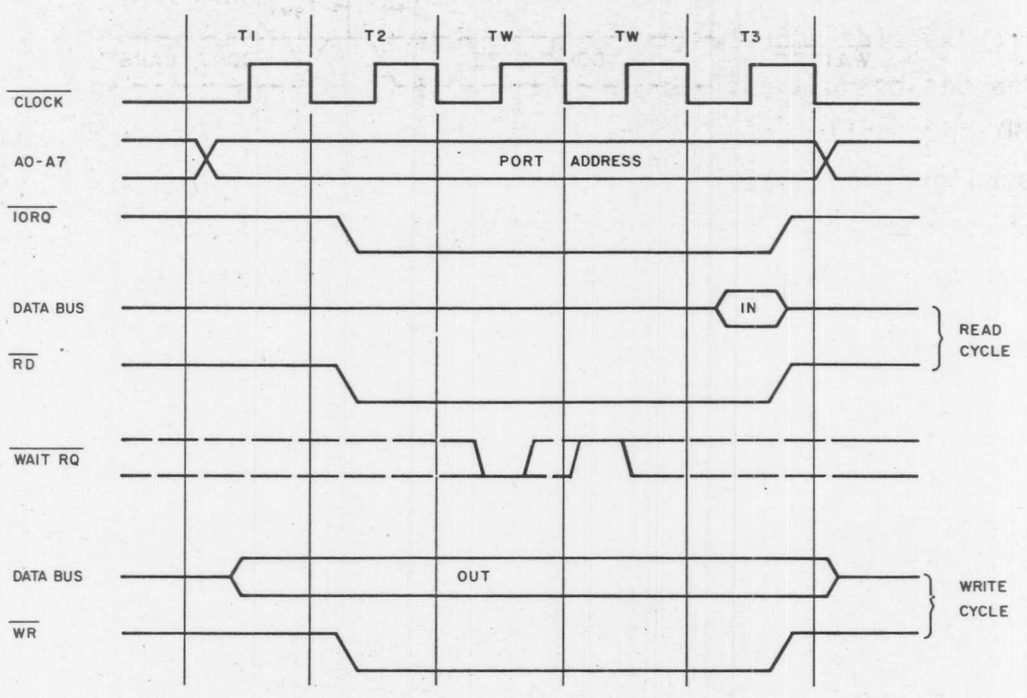


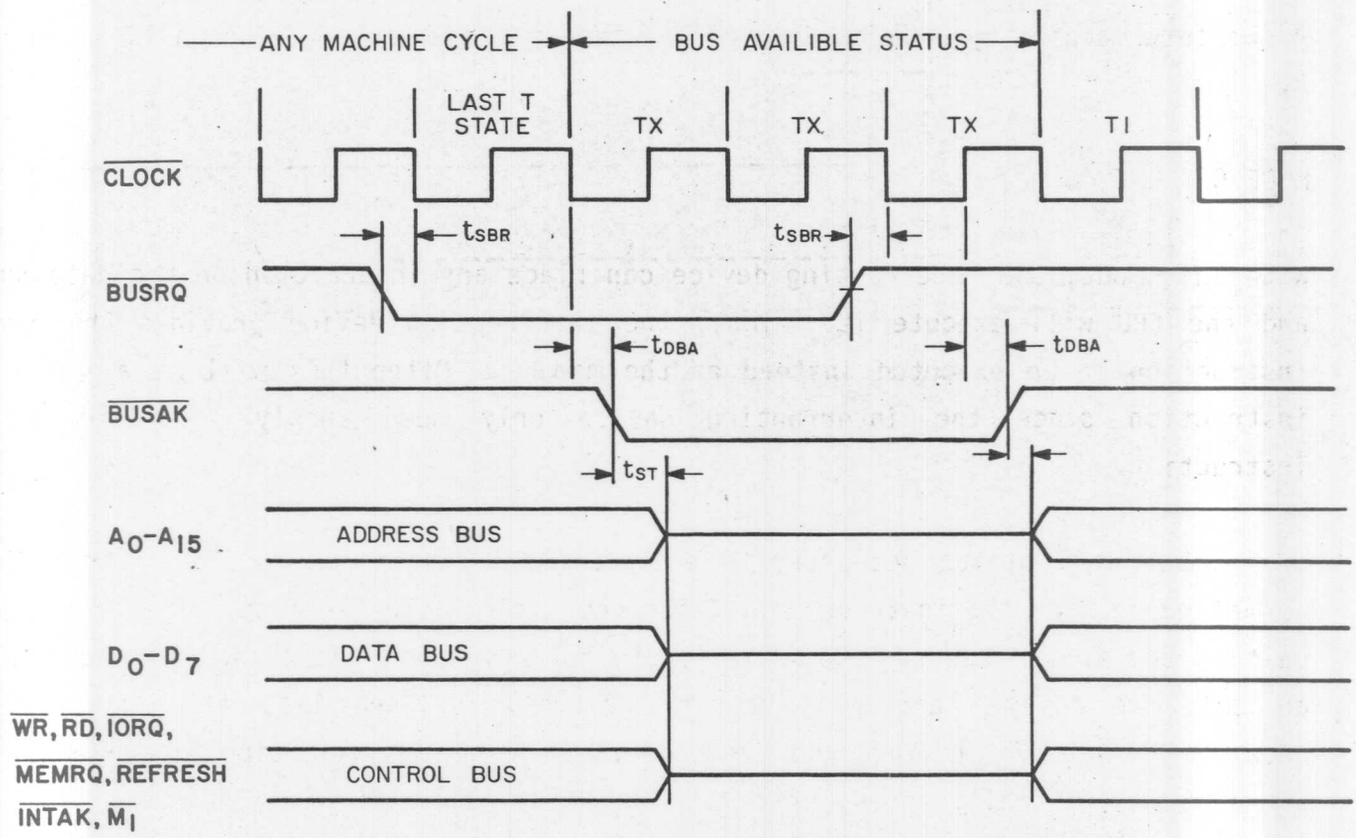
FIGURE 2-9 ADDING WAIT STATES TO I/O CYCLES



BUS REQUEST/ACKNOWLEDGE CYCLE

Figure 2-10 illustrates the timing for a Bus Request/Acknowledge cycle. The /BUSRQ signal is sampled by the CPU with the falling edge of the last clock period of any machine cycle. If the /BUSRQ signal is active, the CPU will set its address, data and tri-state control signals to the high impedance state with the falling edge of the next clock pulse. At that time any external device can control the buses to transfer data between memory and I/O devices. (This is generally known as Direct Memory Access (DMA) using cycle stealing). The maximum time for the CPU to respond to a bus request is the length of a machine cycle and the external controller can maintain control of the bus for as many clock cycles as is desired. Note, however, that if very long DMA cycles are used, and dynamic memories are being used, the external controller must also perform the refresh function. This situation only occurs if very large blocks of data are transferred under DMA control. Also note that during a bus request cycle, the CPU cannot be interrupted by either a /NMIRQ or an /INTRQ signal.

FIGURE 2-10 BUS REQUEST/ACKNOWLEDGE CYCLE



MASKABLE INTERRUPT REQUEST/ACKNOWLEDGE CYCLE

Figures 2-11, 2-12, and 2-13 illustrate the timing associated with an interrupt acknowledge cycle for interrupt modes 0, 1, and 2. The interrupt signal /INTRQ is sampled by the CPU with the falling edge of the last clock at the end of any instruction. The signal will not be accepted if the internal CPU software controlled interrupt enable flip-flop is not set or if the /BUSRQ signal is active. When the signal is accepted a special M1 cycle is generated. During this special M1 cycle the /IORQ signal becomes active (instead of the normal /MEMRQ) to indicate that the interrupting device can place an 8-bit vector on the data bus. Notice that two wait states are automatically added to this cycle. These states are added so that a daisy chain priority interrupt scheme can be easily implemented. The two wait states allow sufficient time for the daisy chain signals to stabilize and identify which I/O device must insert the response vector.

The Z80 is always initialized to interrupt Mode 0 after a power-up or manual reset. Mode 1 and Mode 2, are selected by software instructions.

A complete machine cycle breakdown of the interrupt responses is shown, is Appendix A.

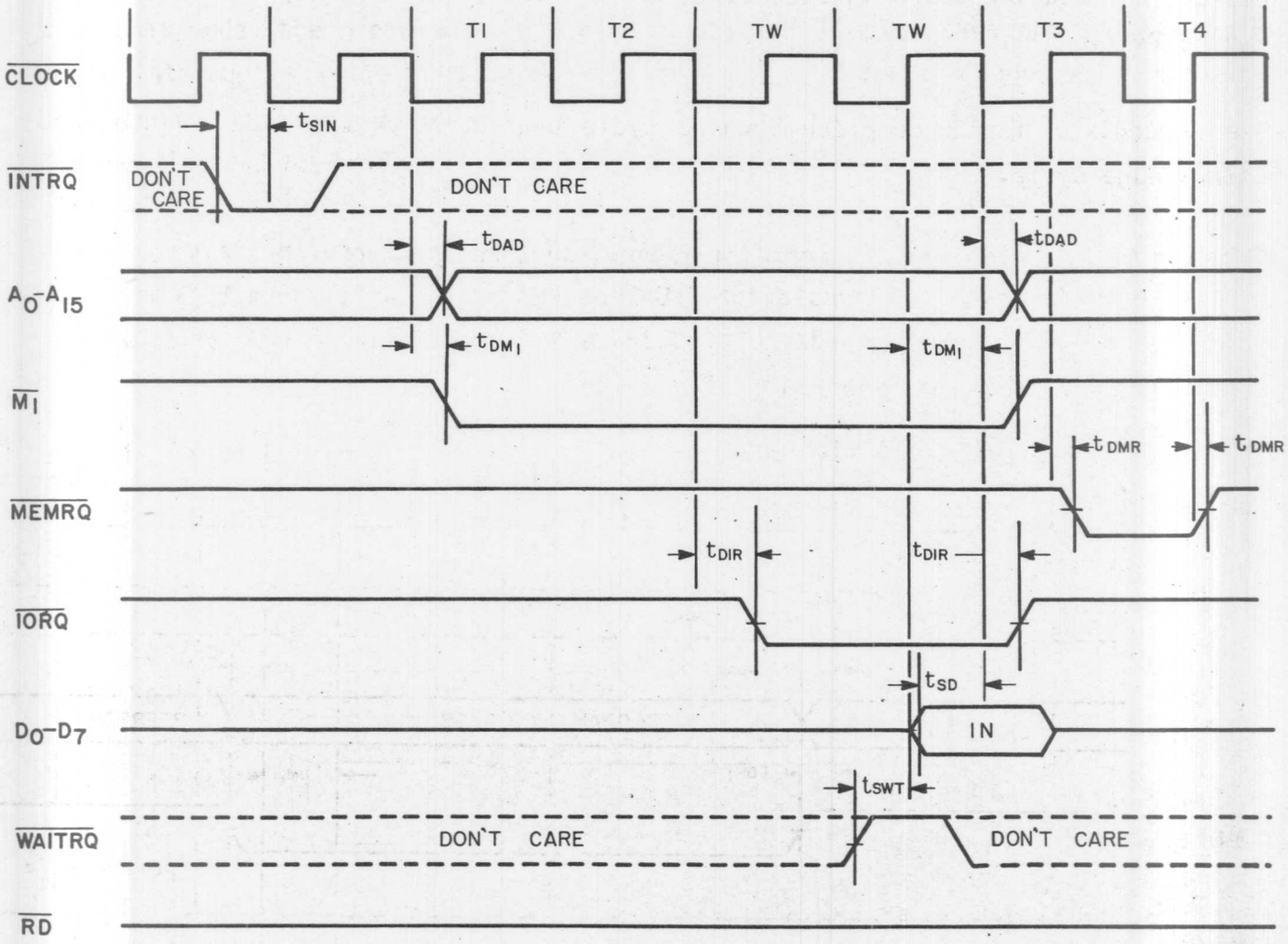
MODE 0

With this mode, the interrupting device can place any instruction on the data bus and the CPU will execute it. Thus, the interrupting device provides the next instruction to be executed instead of the memory. Often this will be a restart instruction since the interrupting device only need supply a single byte instruction.

The number of T states necessary to execute this instruction is 2 more than the normal number for the instruction. This occurs since the CPU automatically adds 2 wait states to an interrupt response cycle to allow sufficient time to implement an external daisy chain for priority control. Figure 2-11 illustrates the detailed timing for an interrupt response. After the application of /RESET the CPU will automatically enter interrupt Mode 0.

See Appendix A for a complete machine cycle breakdown of the Mode 0 interrupt acknowledge cycle.

FIGURE 2-11 MODE 0 INTERRUPT ACKNOWLEDGE CYCLE

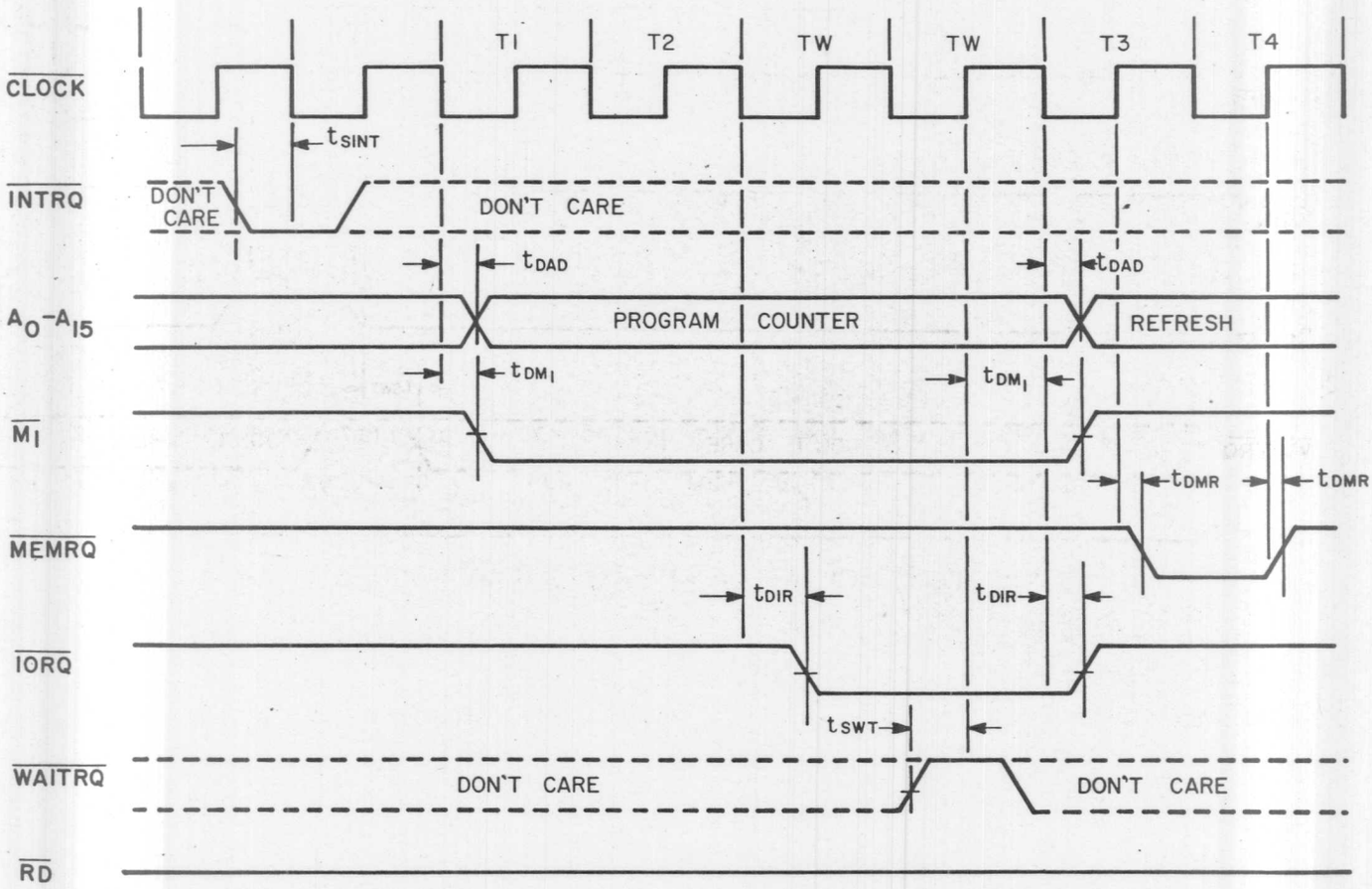


MODE 1

When this mode has been selected by software, the CPU will respond to an interrupt by executing a restart to location 0038H. Thus the response is identical to that for a MODE 0 response, except that interrupt vector is always ignored and a restart to location 0038H is executed. Timing for the MODE 1 response is shown in Figure 2-12.

See Appendix A for a complete machine cycle breakdown of the mode 1 interrupt acknowledge cycle.

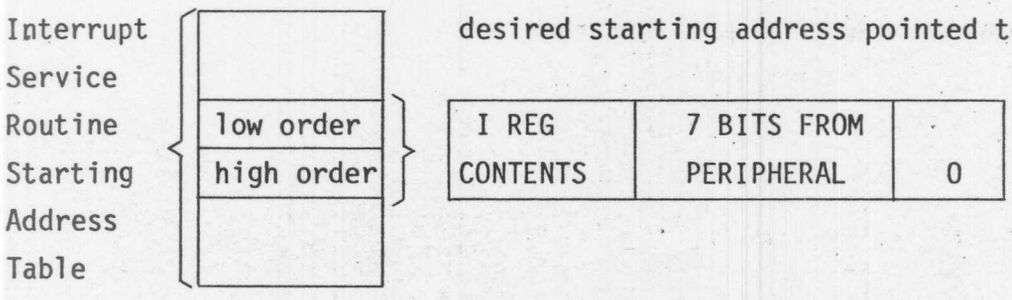
FIGURE 2-12 MODE 1 INTERRUPT ACKNOWLEDGE CYCLE



MODE 2

This mode is the most powerful interrupt response mode. With a single 8-bit byte from the interrupting peripheral, an indirect call can be made to any memory location.

With this mode, the programmer maintains a table of 16 bit starting addresses for every interrupt service routine. This table may be located anywhere in memory. When an interrupt is accepted, a 16 bit pointer must be formed to obtain the desired interrupt service routine starting address from the table. The upper 8 bits of this pointer is formed from the contents of the I register. The I register must have been previously loaded with the desired value by the programmer, i.e. LD I, A. Note that a CPU reset clears the I register so that it is initialized to zero. The lower eight bits of the pointer must be supplied by the interrupting device. Actually, only 7 bits are required since the pointer is used to get two adjacent bytes to from a complete 16 bit service routine starting address and the addresses must always start in even locations.



The first byte in the table is the least significant (low order) portion of the address. The programmer must obviously fill this table in with the desired addresses before any interrupts are to be accepted.

Note that this table can be changed at any time by the programmer (if it is stored in Read/Write Memory) to allow different peripherals to be serviced by different service routines.

Once the interrupting device supplies the lower portion of the pointer, the CPU automatically pushes the program counter onto the stack, obtains the starting address from the table and does a jump to this address. This mode of response requires 19 clock periods to complete (7 to fetch the lower 8 bits from the interrupting device, 6 to save the program counter, and 6 to obtain the jump address.)

Another powerful feature of using MODE 2 interrupts, is the daisy chain priority interrupt structure using PCI and PCO. PCI and PCO form an interrupt daisy chain which prioritizes the interrupting peripherals so that only the highest priority peripherals requesting the interrupt responds with the interrupt vector. Figure 2-13 shows the timing for the Mode 2 interrupt cycle. See Appendix A for the complete machine cycle breakdown of the Mode 2 interrupt cycle.

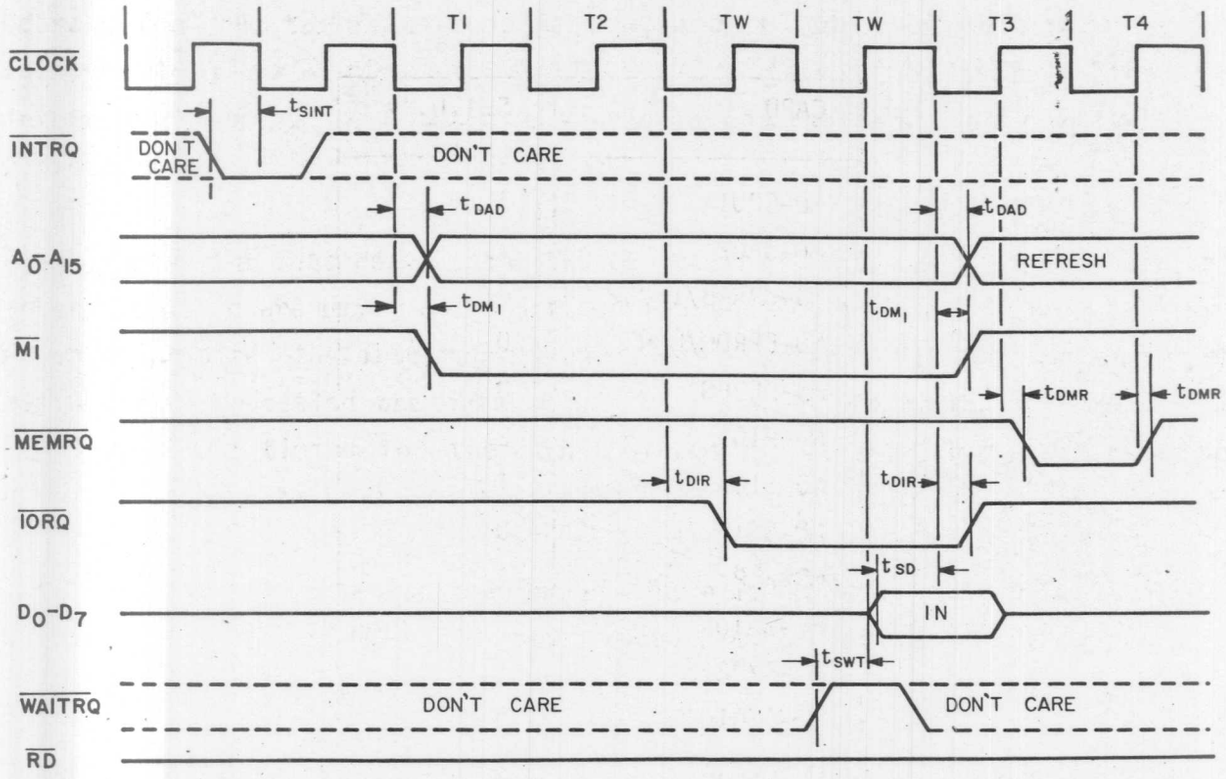
The number of interrupting peripherals in a MD-STD-Z80 system is limited to 5, without the use of MD-INT interrupt expander card. An interrupting peripheral is a Z80-CTC, Z80-PIO, Z80-SIO, or Z80-DMA, and one interrupting peripheral is equal to one S.I.U. or Standard Interrupt Unit. The MD-INT interrupt expander card allows the MD-STD-Z80 system to have up to 40 S.I.U.'s. Table 2-1 gives the S.I.U's for the current Mostek MD cards.

TABLE 2-1
SYSTEM INTERRUPT UNITS

CARD	S.I.U.'s
MD-CPU1	1
MD-CPU2	1
MD-DRAM8/16/32	0
MD-EPROM/UART	0
MD-DEBUG	0
MD-PIO	2
MD-SIO	1
MD-SST	0
MD-FLP	1
MD-MATH	1
MD-A/D8	1
MD-A/D10	0
MD-A/D12	1
MD-D/A8	0
MD-D/A12	0
MD-UMC	0
MD-SRAM4/8/16	0
MD-EPROM	0
MD-INT	1
MD-SC/D	1

NOTE: The number of S.I.U's in a MD-STD-Z80 is limited to 5 without the use of the MD-INT interrupt expander card.

FIGURE 2-13 MODE 2 INTERRUPT ACKNOWLEDGE CYCLE



NONMASKABLE INTERRUPT RESPONSE

Figure 2-14 illustrates the request/acknowledge cycle for the nonmaskable interrupt. A pulse on the /NMIRQ input sets an internal /NMIRQ latch which is tested by the CPU at the end of every instruction. The /NMIRQ latch is sampled at the same time as the interrupt line, but this line has priority over the normal interrupt and it can not be disabled under software control. Its usual function is to provide immediate response to important signals such as an impending power failure. The CPU response to a nonmaskable interrupt is similar to a normal memory read operation. The only difference being that the content of the data bus is ignored while the processor automatically stores the PC in the external stack and jumps to location 0066_H. The service routine for the nonmaskable interrupt must begin at this location if this interrupt is used.

FIGURE 2-14 NONMASKABLE INTERRUPT RESPONSE

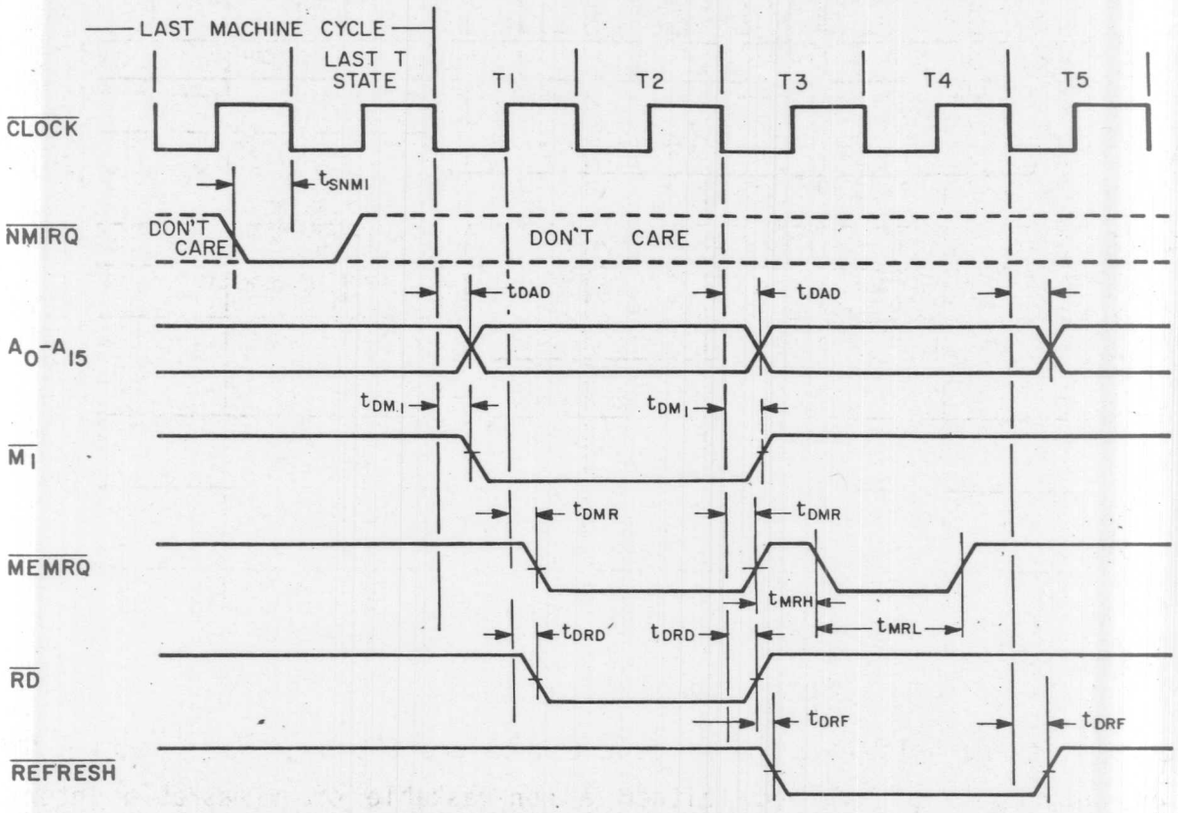
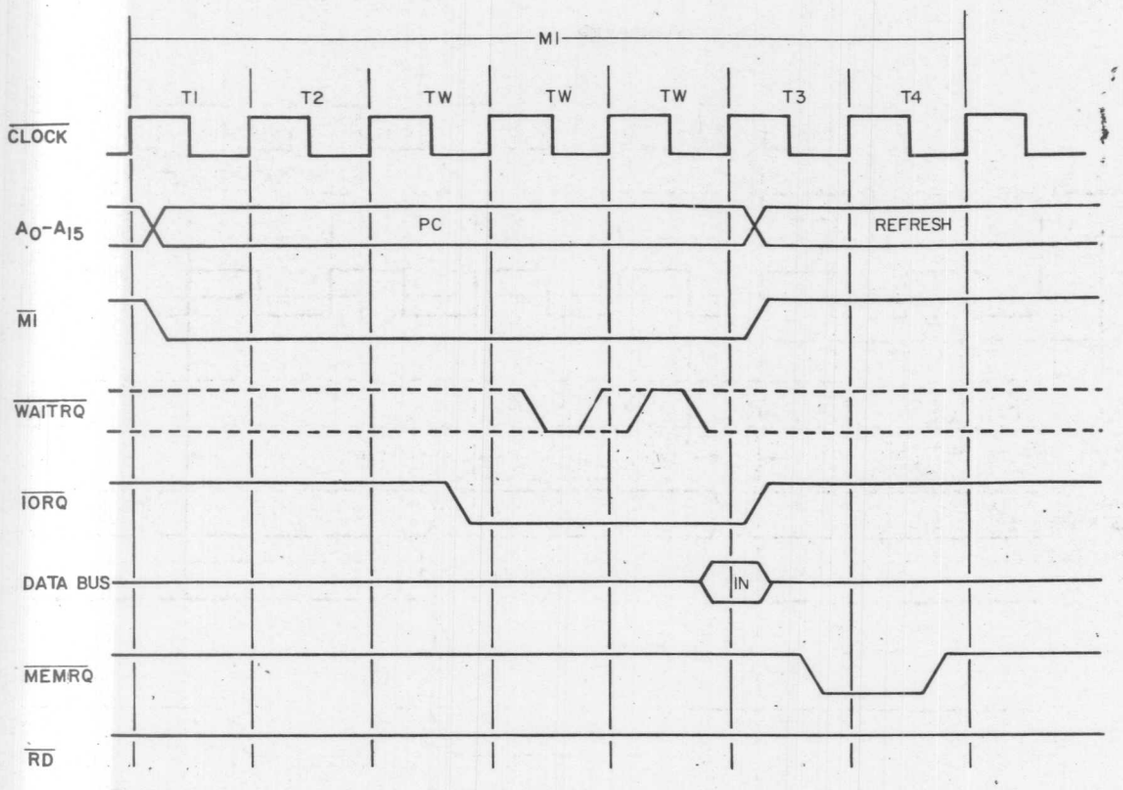


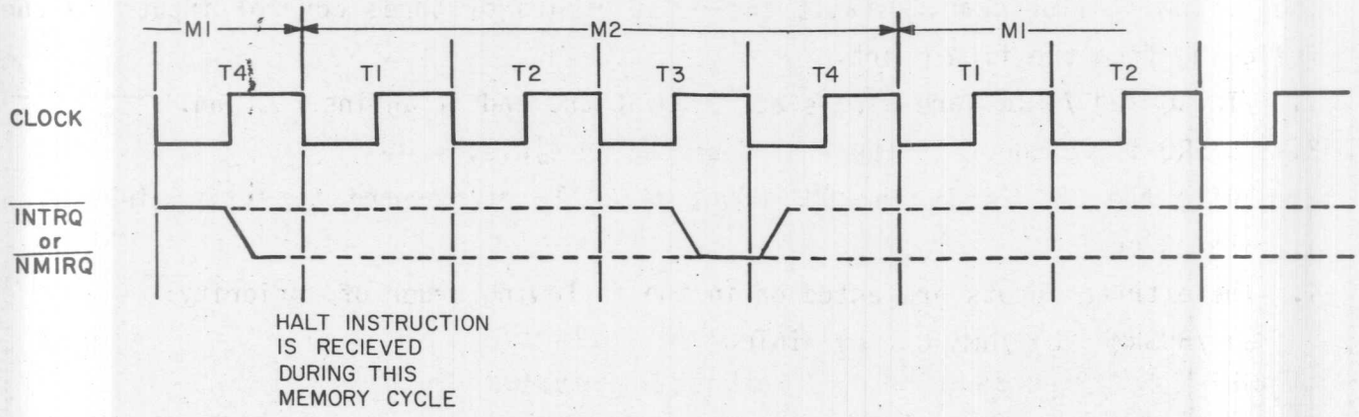
FIGURE 2-15 INTERRUPT ACKNOWLEDGE WITH WAIT STATES



HALT EXIT

Whenever a software halt instruction is executed the CPU begins executing NOP's until an interrupt is received (either a non-maskable or a maskable interrupt while the interrupt flip-flop is enabled). The two interrupt lines are sampled with the falling clock edge during each T4 state as shown in Figure 2-16. If a non-maskable interrupt has been received or a maskable interrupt has been received and the interrupt enable flip-flop is set, then the halt state will be exited on the next falling clock edge. The following cycle will then be an interrupt acknowledge cycle corresponding to the type of interrupt that was received. If both are received at this time, then the non maskable will be aknowledged since it was highest priority. The purpose of executing NOP instructions while in the halt state is to keep the memory refresh signals active. Each cycle in the halt state is a normal M1 (fetch) cycle except that the data received from the memory is ignored and a NOP instruction is forced internally to the CPU.

FIGURE 2-16 HALT EXIT

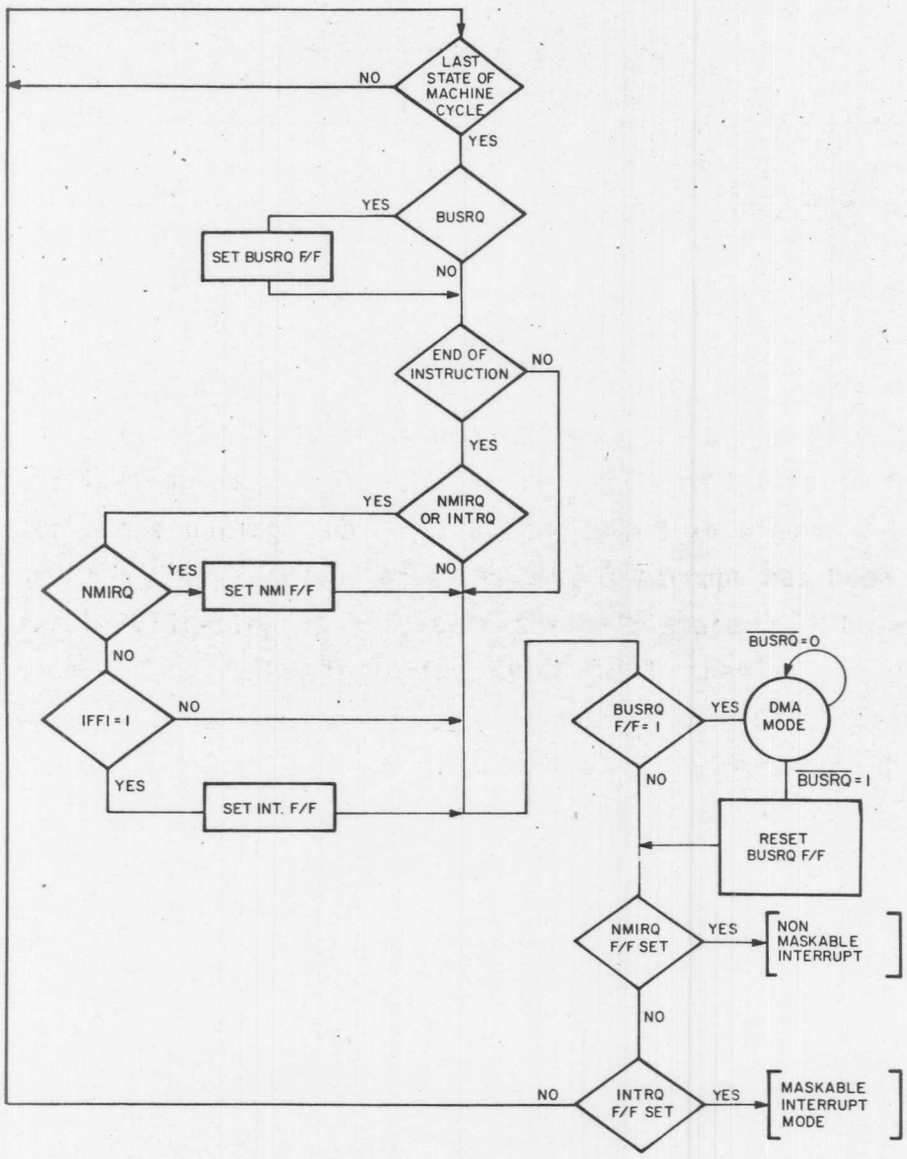


INTERRELATIONSHIP OF /INTRQ OR /NMIRQ AND /BUSRQ

The following flow chart details the relationship of three control inputs to the following from the flow chart.

1. /INTRQ and /NMIRQ are always acted on at the end of an instruction.
2. /BUSRQ is acted on at the end of a machine cycle.
3. While the CPU is in the DMA MODE, it will not respond to active inputs on /INTRQ or /NMIRQ
4. These three inputs are acted on in the following order of priority:
   a) /BUSRQ b) /NMIRQ c) /INTRQ

FIGURE 2-17 CPU INTERRUPT SEQUENCE



MD-STD-Z80 BUS TIMING
2.5 MHZ

BUS SIGNAL	SYMBOL	DESCRIPTION	MIN	MAX	UNITS
/CLOCK	t_c	Clock Period		400NS	
A_0-A_{15}	t_{DAD}	Address Output Delay		180	NS
	t_{ACM}	Address Stable Prior to /MEMRQ	125		NS
	t_{ACI}	Address Stable Prior to /IORQ, /RD /WR (I/O CYCLE)	320		NS
D_0-D_7	t_{SDO}	Data setup time during M1 cycle	70		NS
	t_{SDR}	Data setup time during memory read cycle, I/O read cycle or interrupt acknowledge cycle.	80		NS
	t_{DCM}	Data stable prior to /WR (Memory Write)	190		NS
	t_{DD}	Data output delay		250	NS
	t_{DCI}	Data stable prior to /WR (I/O write)	0		NS
	t_{CDF}	Data stable from /WR	120		NS
/MEMRQ	t_{DMR}	/MEMRQ delay from /CLOCK		120	NS
	t_{MRH}	/MEMRQ pulse width high	170		NS
	t_{MRL}	/MEMRQ pulse width low	360		

MD-STD-Z80 BUS TIMING
2.5 MHZ

BUS SIGNAL	SYMBOL	DESCRIPTION	MIN	MAX	UNITS
/IORQ	t_{DIR}	/IORQ delay from /CLOCK		110	NS
/RD	t_{DRD}	/RD delay from /CLOCK		120	NS
/WR	$t_{\overline{DWR}}$	/WR delay from /CLOCK		110	NS
/M1	t_{DMI}	/M1 delay from /CLOCK		150	NS
/REFRESH	t_{DRF}	/REFRESH delay from /CLOCK		180	NS
/WAITRQ	t_{SWT}	/WAITRQ setup time	110		NS
/INTRQ	t_{SINT}	/INTRQ setup time	100		NS
/NMIRQ	t_{WNMI}	/NMIRQ pulse width time	100		NS
/BUSRQ	t_{SBR}	/BUSRQ setup time	100		NS

MD-STD-Z80 BUS TIMING
2.5 MHZ

BUS SIGNAL	SYMBOL	DESCRIPTION	MIN	MAX	UNITS
/BUSAK	t_{DBA}	/BUSAK delay from /CLOCK		140	NS
	t_{TS}	Tri-state buffer delay from BUSAK for A_0-A_{15}, D_0-D_7, /MEMRQ, /IORQ, /M1, /RD, /WR, /REFRESH, and /INTAK		20	NS

MD-STD-Z80 BUS TIMING
4.0 MHZ

BUS SIGNAL	SYMBOL	DESCRIPTION	MIN	MAX	UNITS
/CLOCK	t_c	Clock Period		250	NS
A_0-A_{15}	t_{DAD}	Address Output Delay		130	NS
	t_{ACM}	Address Stable Prior to /MEMRQ	75		NS
	t_{ACI}	Address stable Prior to /IORQ, /RD /WR (I/O CYCLE)	180		NS
D_0-D_7	t_{SDO}	Data setup time during /M1 cycle	70		NS
	t_{SDR}	Data setup time during memory read cycle, I/O read cycle or interrupt acknowledge cycle.	80		NS
	t_{DCM}	Data stable prior to /WR (Memory Write)	80		NS
	t_{DD}	Data output delay	155		NS
	t_{DCI}	Data stable prior to /WR (I/O write)	-30		NS
	t_{CDF}	Data stable from /WR	70		NS
/MEMRQ	t_{DMR}	/MEMRQ delay from /CLOCK		95	NS
	t_{MRH}	/MEMRQ pulse width high	105		NS
	t_{MRL}	/MEMRQ pulse width low	220		NS

MD-STD-Z80 BUS TIMING
4.0 MHZ

BUS SIGNAL	SYMBOL	DESCRIPTION	MIN	MAX	UNITS
/IORQ	t_{DIR}	/IORQ delay from /CLOCK		85	NS
/RD	t_{DRD}	/RD delay from /CLOCK		95	NS
/WR	t_{DWR}	/WR delay from /CLOCK		75	NS
/M1	t_{DMI}	/M1 delay from /CLOCK		110	NS
/REFRESH	t_{DRF}	/REFRESH delay from /CLOCK		130	NS
/WAITRQ	t_{SWT}	/WAITRQ setup time	85		NS
/INTRQ	t_{SINT}	/INTRQ setup time	95		NS
/NMIRQ	t_{SNMI}	/NMIRQ setup time	85		
/BUSRQ	t_{SBR}	/BUSRQ setup time	55		NS
/BUSAK	t_{DBA}	/BUSAK delay from /CLOCK		110	NS
	t_{TS}	Tri-state buffer delay from /BUSAK for A_0-A_{15}, D_0-D_7, /MEMRQ, IORQ, /M1 /RD, /WR, /REFRESH, and/INTAK.		25	NS

APPENDIX A

Z80 INSTRUCTION BREAKPOINT BY MACHINE CYCLE

LEGEND

IO	– Internal CPU Operation	ODL	– Operand Data Read of Low Byte
MR	– Memory Read	PR	– Port Read
MRH	– Memory Read of High Byte	PW	– Port Write
MRL	– Memory Read of Low Byte	SRH	– Stack Read of High Byte
MW	– Memory Write	SRL	– Stack Read of Low Byte
MWH	– Memory Write of High Byte	SWH	– Stack Write of High Byte
MWL	– Memory Write of Low Byte	SWL	– Stack Write of Low Byte
OCF	– Op Code Fetch	( )	– Number of T-States in that Machine Cycle
ODH	– Operand Data Read of High Byte		

Z80 INSTRUCTION BREAKDOWN BY MACHINE CODE

INSTRUCTION TYPE	BYTES	MACHINE CYCLE M1	M2	M3	M4	M5
LD r, s	1	OCF (4)				
LD r, n	2	OCF (4)	OD (3)			
LD r, (HL)	1	OCF (4)	MR (3)			
LD (HL), r		OCF (4)	MW (3)			
LD r, (IX+d)	3	OCF (4)/OCF (4)	OD (3)	IO (5)	MR (3)	
LD (IX+d), r		OCF (4)/OCF (4)	OD (3)	IO (5)	MW (3)	
LD (HL), n	2	OCF (4)	OD (3)	MW (3)		
LD A, ($^{BC}_{DE}$)	1	OCF (4)	MR (3)			
LD ($^{BC}_{DE}$), A		OCF (4)	MW (3)			
LD A, (nn)	3	OCF (4)	ODL (3)	ODH (3)	MR (3)	
LD (nn), A		OCF (4)	ODL (3)	ODH (3)	MW (3)	
LD A, $^{I}_{R}$	2	OCF (4)/OCF(5)				
LD $^{I}_{R}$, A						
LD dd, nn	3	OCF (4)	ODL (3)	ODH (3)		
LD IX, nn	4	OCF (4)/OCF (4)	ODL (3)	ODH (3)		
LD HL, (nn)	3	OCF (4)	ODL (3)	ODH (3)	MRL (3)	MRH (3)
LD (nn), HL		OCF (4)	ODL (3)	ODH (3)	MWL (3)	MWH (3)
LD dd, (nn)	4	OCF (4)/OCF (4)	ODL (3)	ODH (3)	MRL (3)	MRH (3)
LD (nn), dd		OCF (4)/OCF (4)	ODL (3)	ODH (3)	MWL (3)	MWH (3)
LD IX, (nn)		OCF (4)/OCF (4)	ODL (3)	ODH (3)	MRL (3)	MRH (3)
LD (nn), IX		OCF (4)/OCF (4)	ODL (3)	ODH (3)	MWL (3)	MWH (3)
LD SP, HL	1	OCF (6)				
LD SP, IX	2	OCF (4)/OCF (6)				
PUSH qq	1	OCF (5) SP-1 →	SWH (3) SP-1 →	SWL (3)		
PUSH IX	2	OCF (4)/OCF (5) SP-1 →	SWH (3) SP-1 →	SWL (3)		
POP qq	1	OCF (4)	SRH (3) SP+1 →	SRL (3) SP+1 →		
POP IX	2	OCF (4)/OCF (4)	SRH (3) SP+1 →	SRL (3) SP+1 →		
EX DE, HL	1	OCF (4)				
EX AF, AF'	1	OCF (4)				

INSTRUCTION TYPE	BYTES	MACHINE CYCLE M1	M2	M3	M4	M5
EXX	1	OCF (4)				
EX (SP), HL	1	OCF (4)	SRL (3)	SRH (4)	SWH (3)	SWL (5)
			SP+1 →		SP-1 →	
EX (SP), IX	2	OCF (4)/OCF (4)	SRL (3)	SRH (4)	SWH (3)	SWL (5)
			SP+1 →		SP-1 →	
LDI LDD CPI CPD	2	OCF (4)/OCF (4)	MR (3)	MW (5)		
LDIR LDDR CPIR CPDR	2	OCF (4)/OCF (4)	MR (3)	MW (5)	IO (5)* *only if BC ≠ 0	
ALU A, r ADD ADC SUB SBC AND OR XOR CP	1	OCF (4)				
ALU A, n	2	OCF (4)	OD (3)			
ALU A, (HL)	1	OCF (4)	MR (3)			
ALU A, (IX+d)	3	OCF (4)/OCF (4)	OD (3)	IO (5)	MR (3)	
DEC INC r	1	OCF (4)				
DEC INC (HL)	1	OCF (4)	MR (4)	MW (3)		
DEC INC (IX+D)	2	OCF (4)/OCF (4)	OD (3)	IO (5)	MR (4)	MW (3)
DAA CPL CCF SCF NOP HALT DI EI	1	OCF (4)				
NEG IMO IM1 IM2	2	OCF (4)/OCF (4)				

INSTRUCTION TYPE	BYTES	MACHINE CYCLE M1	M2	M3	M4	M5
ADD HL, ss	1	OCF (4)	IO (4)	IO (3)		
ADC HL, ss SBC HL, ss ADD IX, pp	2	OCF (4)/OCF (4)	IO (4)	IO (3)		
INC ss DEC ss	1	OCF (6)				
DEC IX INC IX	2	OCF (4)/OCF (6)				
RLCA RLA RRCA RRA	1	OCF (4)				
RLC r RL RRC RR SLA SRA SRL	2	OCF (4)/OCF (4)				
RLC (HL) RL RRC RR SLA SRA SRL	2	OCF (4)/OCF (4)	MR (4)	MW (3)		
RLC (IX+d) RL RRC RR SLA SRA SRL	4	OCF (4)/OCF (4)	OD (3)	IO (5)	MR (4)	MW (3)
RLD RRD	2	OCF (4)/OCF (4)	MR (3)	IO (4)	MW (3)	
BIT b, r SET RES	2	OCF (4)/OCF (4)				

INSTRUCTION TYPE	BYTES	MACHINE CYCLE M1	M2	M3	M4	M5
BIT b, (HL)	2	OCF (4)/OCF (4)	MR (4)			
SET b, (HL) RES	2	OCF (4)/OCF (4)	MR (4)	MW (3)		
BIT b, (IX+d)	4	OCF (4)/OCF (4)	OD (3)	IO (5)	MR (4)	
SET b, (IX+d) RES	4	OCF (4)/OCF (4)	OD (3)	IO (5)	MR (4)	MW (3)
JP nn JP cc, nn	3	OCF (4)	ODL (3)	ODH (3)		
JR e	2	OCF (4)	OD (3)	IO (5)		
JR C, e JR NC, e JR Z, e JR NZ, e	2	OCF (4)	OD (3)	IO (5)* * If condition is met		
JP (HL)	1	OCF (4)				
JP (IX)	2	OCF (4)/OCF (4)				
DJNZ, e	2	OCF (5)	OD (3)	IO (5)* * If B≠ 0		
CALL nn CALL cc, nn cc true	3	OCF (4)	ODL (3)	ODH (4) SP-1 →	SWH (3) SP-1 →	SWL (3)
CALL cc, nn cc false	3	OCF (4)	ODL (3)	ODH (3)		
RET	1	OCF (4)	SRL (3) SP+1 →	SRH (3)	SP+1 →	
RET cc	1	OCF (5)	SRL (3)* * If cc is true SP+1 →	SRH (3)*	SP+1 →	
RETI RETN	2	OCF (4)/OCF (4)	SRL (3) SP+1 →	SRH (3)	SP+1 →	
RST p	1	OCF (5) SP-1 →	SWH (3) SP-1 →	SWL (3)		

INSTRUCTION TYPE	BYTES	MACHINE CYCLE M1	M2	M3	M4	M5
IN A, (n)	2	OCF (4)	OD (3)	PR (4)		
IN r, (c)	2	OCF (4)/OCF (4)	PR (4)			
INI IND	2	OCF (4)/OCF (5)	PR (4)	MW (3)		
INIR INDR	2	OCF (4)/OCF (5)	PR (4)	MW (3)	IO (5)	
OUT (n) , A	2	OCF (4)	OD (3)	PW (4)		
OUT (C), r	2	OCF (4)/OCF (4)	PW (4)			
OUTI OUTD	2	OCF (4)/OCF (5)	MR (3)	PW (4)		
OTIR OTDR	2	OCF (4)/OCF (5)	MR (3)	PW (4)	IO (5)	
INTERRUPTS						
NMI	–	OCF (5) *	SWH (3)	SWL (3)	*Op Code Ignored	
		SP-1 →	SP-1 →			
INT						
MODE 0	–	INTA (6) (CALL INSERTED)	ODL (3)	ODH (4)	SWH (3)	SWL (3)
				SP-1 →	SP-1 →	
	–	INTA (6) (RST INSERTED)	SWH (3)	SWL (3)		
		SP-1 →	SP-1 →			
MODE 1		INTA (7) (RST 38H INTERNAL)	SWH (3)	SWL (3)		
		SP-1 →	SP-1 →			
MODE 2	–	INTA (7) (VECTOR SUPPLIED)	SWH (3)	SWL (3)	MRL (3)	MRH (3)
		SP-1 →	SP-1 →			